MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY & MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
NAAC Accredited Institution with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

Department of Computer Science and Engineering

MAGPIE: A demonstration of symmetric encryption

BATCH - 2

Amulya Reddy Y -217Y1A0572

Srinivas Rao T -217Y1A05C0

GUIDE

Mr. T. S. Srinivas
Associate Professor

HOD

Dr . K. Abdul Basith

Index

Abstract

The project "**Magpie**" addresses this need by demonstrating encryption through a Python-based application. This project is important as it provides both a command-line interface (**CLI**) and a graphical user interface (**GUI**), making cryptographic principles accessible and interactive for users.

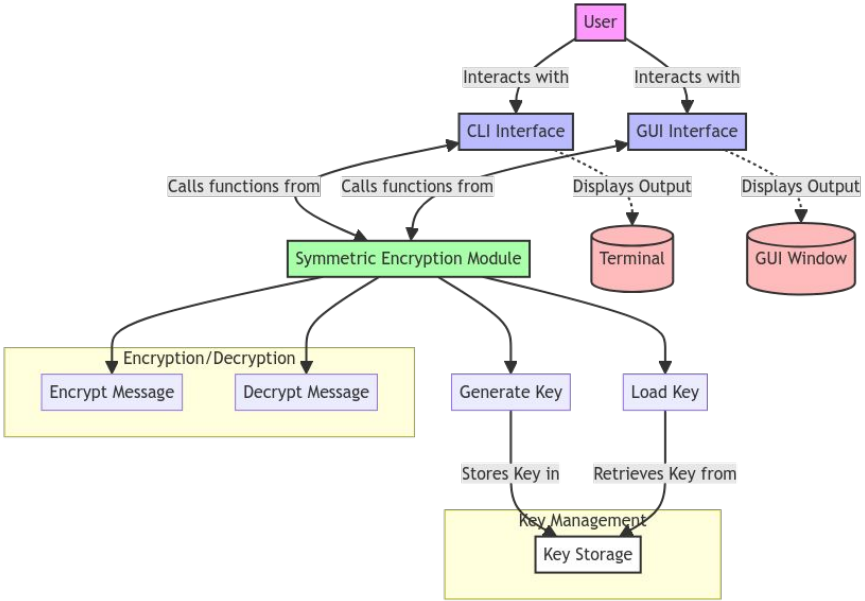
The primary objective of the Magpie project is to implement and demonstrate the fundamental principles of cryptography and information security. It later aims to offer **Image encryption and decryption** using a super key by implementing **SHA-512**. Magpie employs Python as the core programming language, integrating the cryptography library for implementing symmetric encryption using the hashing through **SHA-256**. The methodology involves key generation, encryption, decryption, and error handling to ensure secure and efficient processing of text messages.

The project is anticipated to effectively demonstrate the concepts of symmetric encryption and hashing, providing security and users with a practical understanding of cryptographic principles.

Magpie has the potential to significantly impact the educational domain by serving as a valuable learning tool for cryptography and information security. It simplifies complex cryptographic concepts, making them accessible to a broader audience.

Objectives

1. ***Key Management:***
    a. **`generate_key ()`**: Generates a new symmetric encryption key and saves it to a file named ‘key.key’.
    b. **`load_key ()`**: Loads the encryption key from the ‘key.key’ file. If the file does not exist, it generates a new key.
2. ***Encryption:*** Encrypts a message using symmetric encryption with the provided key.
3. ***Decryption:*** Decrypts an encrypted message using the provided key. It handles potential errors such as invalid tokens or bin ascii errors.
4. ***Error Handling:*** The module handles errors such as missing key files and invalid tokens during decryption. It raises custom exceptions (`Key NotFoundError`) to notify users of key-related issues.

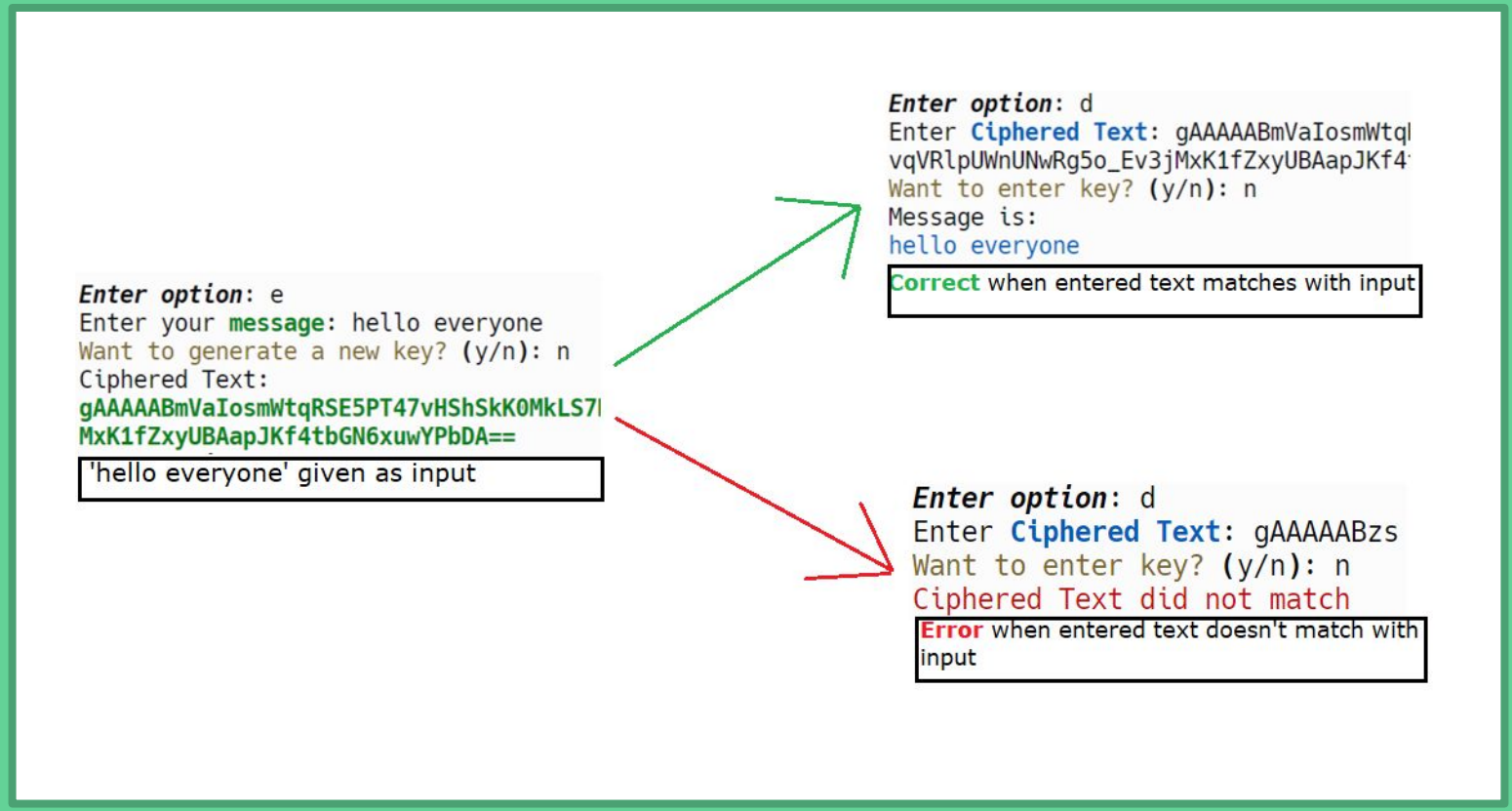


System Architecture

Proposed System

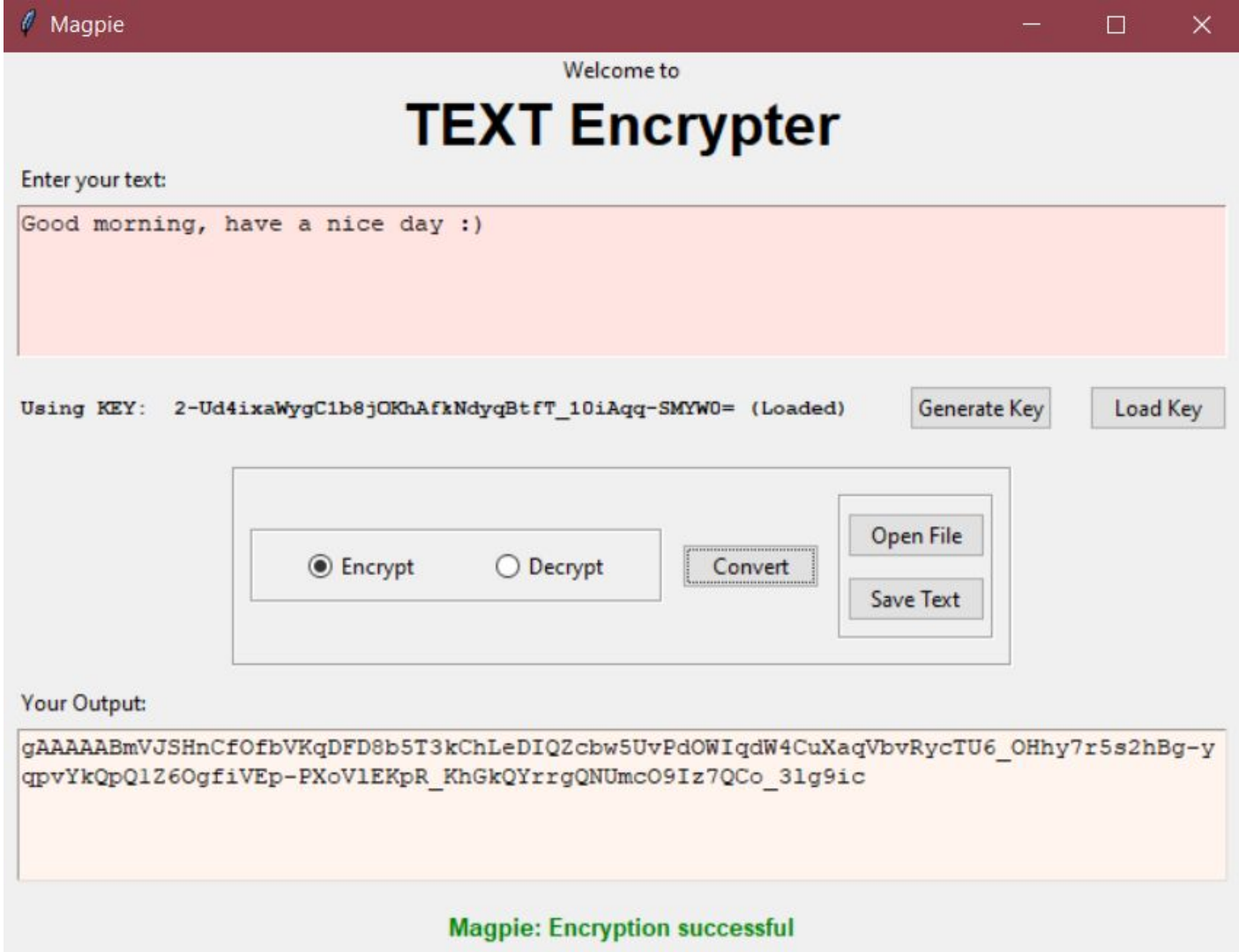
1. **User-friendly GUI:** The encryption system offers an intuitive graphical user interface (GUI) that simplifies the encryption and decryption process, making it accessible even for users without technical expertise.
2. **Hash Libraries (SHA-256) and Fernet Cryptography:** The system utilizes SHA-256 for hashing, ensuring data integrity, and Fernet cryptography for encryption, providing strong and reliable security measures to protect sensitive information.
3. **Error Logging for Debugging and Auditing:** The system incorporates robust error logging mechanisms, enabling developers to troubleshoot issues efficiently and maintain a detailed audit trail for security and compliance purposes.

Output


Enter option: e
Enter your message: hello everyone
Want to generate a new key? (y/n): n
Ciphered Text:
gAAAAABmVaIosmWtqRSE5PT47vHShSkK0MkLS7
MxK1fZxyUBAapJKf4tbGN6xuwYPbDA==
'hello everyone' given as input
Enter option: d
Enter Ciphered Text: gAAAAABmVaIosmWtq
vqVRlpUWnUNwRg5o_Ev3jMxK1fZxyUBAapJKf4
Want to enter key? (y/n): n
Message is:
hello everyone
Correct when entered text matches with input
Enter option: d
Enter Ciphered Text: gAAAAABzs
Want to enter key? (y/n): n
Ciphered Text did not match
Error when entered text doesn't match with input

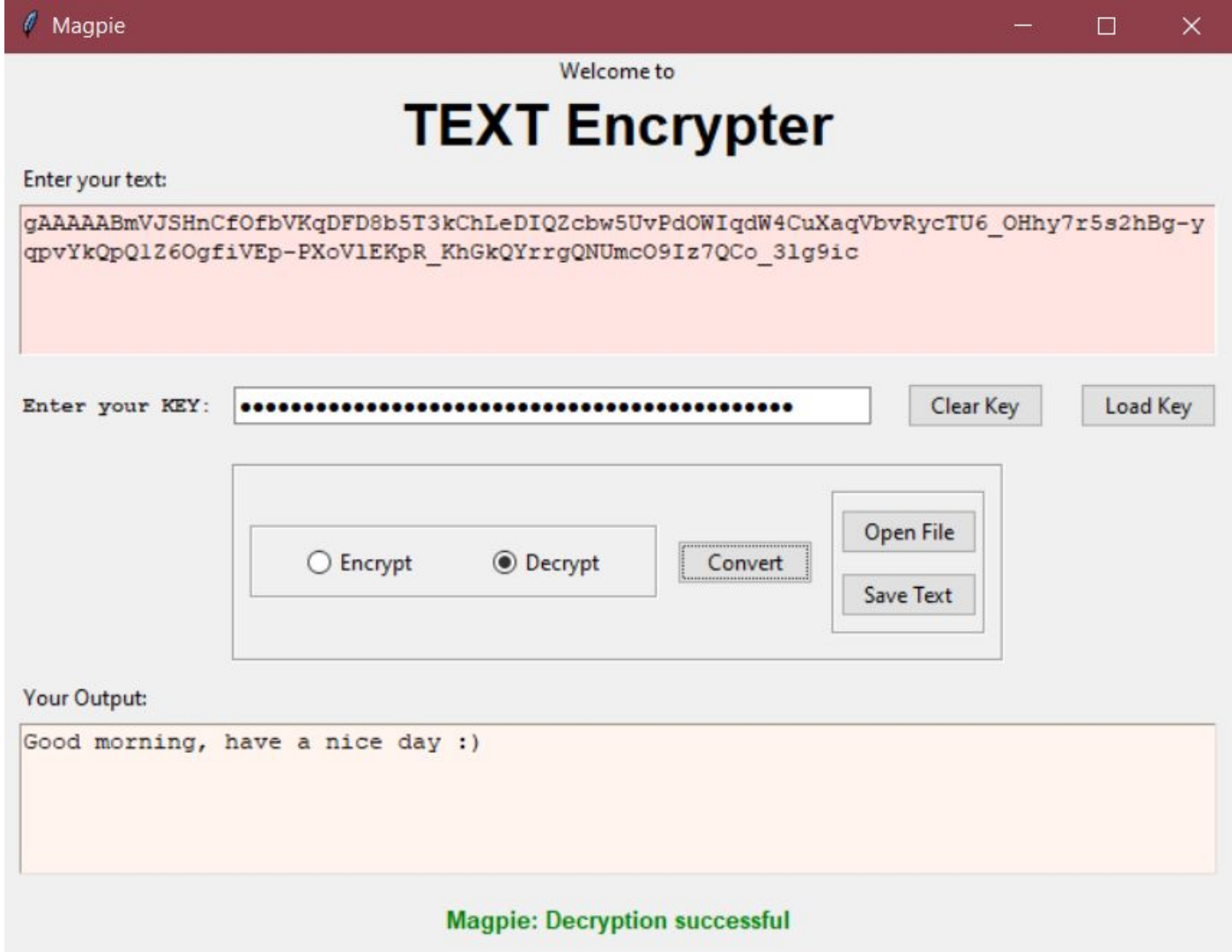
Output

Encryption using GUI (Graphical User Interface)



Output

Decryption using GUI (Graphical User Interface)



Conclusion

The Secure Message Encryption and Decryption System represents a fundamental building block for implementing secure communication and data protection in Python-based applications.

By prioritizing key management, robust encryption algorithms, and error handling mechanisms, the system offers a reliable solution for safeguarding sensitive information against unauthorized access and tampering.

References

1. Cryptography Library
   - Cryptography. (n.d.). cryptography.io
2. Python Standard Library
   - os Module: Python Software Foundation. (n.d.). os
   - binascii Module: Python Software Foundation. (n.d.). binascii
3. PEP 484 – Type Hints
   - van Rossum, G., & Lehtosalo, J. (2014). PEP 484
4. CLI.py
   - Rich Library Documentation
   - Python Match Statement
5. GUI.py
   - Tkinter Documentation